Bài tập về nhà

(1) Xác định kích thước dữ liệu vào và bậc tăng trưởng độ phức tạp thuật toán.

Câu 3: Kiểm tra tìm kiếm tuần tự trong danh sách để tìm kiếm 1 khóa trong danh sách:

- Trường hợp xấu nhất: Khóa không tồn tại trong danh sách hoặc là xuất hiện ở cuối danh sách => phải duyệt qua tất cả phàn tử trong danh sách => số lần so sánh là n => Độ phức tạp là O(n).

- Trường hợp trung bình: Khóa xuất hiện ngẫu nhiên trong danh sách => duyệt khoảng nửa số phần tử trước khi tìm thấy khóa tìm kiếm => số lần so sánh trung bình là n/2 => Độ phức tạp là O(n/2) = O(n).

- Trường hợp tốt nhất: Khóa xuất hiện ở vị trí đầu danh sách => chỉ cần 1 lần so sánh => Độ phức tạp là O(1).

Câu 4:

a)

TH tốt nhất: 2 lần lấy (cùng màu, cùng chiều)

TH xấu nhất: 12 lần lấy (11 lần lấy cùng chiều, lần thứ 12 lấy khác chiều)

b) 5 đôi tất phân biệt:

TH tốt nhấtL 4 đôi hoàn chỉnh

TH xấu nhất: 3 đôi hoàn chỉnh

=> Số cách lấy 2 chiếc trong 10 chiếc: $C^2_{10} = 45$

TH tốt nhất: 2 chiếc cùng 1 đôi: 5 cách => xác suất $\frac{5}{45} = \frac{1}{9}$.

TH xấu nhất: 2 chiếc khác đôi: $1 - \frac{1}{9} = \frac{8}{9}$.

TH số cặp trung bình = $4 * \frac{1}{9} + 3 * \frac{8}{9} = \frac{28}{9}$

Câu 8: n = f(n) => f(3n) = 3n

a) nlog2n = 3nlog(6n) => tăng gấp 3log2(6) khi n tiến đến vô cùng
b) n => 3n => tăng gấp 3 lần
c) $n^3 \Rightarrow (3n)^3 = 27n^3$ => tăng gấp 27 lần
d) $n^2 \Rightarrow (3n)^2 = 9n^2$ => tăng gấp 9 lần
e) n! = 3n! => (3n)(3n-1)...(n+1)n! => tăng rất nhanh gấp (3n)(3n-1)...(n+1) lần
f) $2^n \Rightarrow 2^{3n} \Rightarrow 8^n$ => tăng gấp 8 lần

Câu 9: So sánh tốc độ tăng trưởng:

a) $n(n + 1)$ và $2000n^2$ có cùng tốc độ tăng trưởng $O(n^2)$ vì $n(n+1) = n^2 + n <= 2n^2$ => với $n >=1$ có tốc độ tăng trưởng là $O(n^2)$, và $2000n^2 >= n^2 + n$ với mọi n, có tốc độ tăng trưởng là $O(n^2)$.
b) $100n^2$ có tốc độ tăng trưởng thấp hơn với $0.01n^3$ vì $100n^2$ có tốc độ tăng trưởng là $O(n^2)$ còn $0.01n^3$ có tốc độ tăng trưởng là $O(n^3)$.
c) $\log_2 n$ có cùng tốc độ tăng trưởng với ln(n) vì $\log_2 n = (\ln(n)) / (\ln(2))$ mà ln(2) là 1 hằng số => $\log_2 n$ và ln(n) có tốc độ tăng trưởng O(ln(n)) .
d) $\log^2{}_2(n)$ có tốc độ tăng trưởng cao hơn $\log_2(n^2)$ vì $\log^2{}_2(n) = n$ có tốc độ tăng trưởng O(n), $\log_2(n^2) = 2\log_2(n)$ có tốc độ tăng trưởng O(logn)
e) 2^{n-1} có tốc độ tăng trưởng thấp hơn 2^n vì $2^{n-1} = \frac{1}{2} * 2^n$.
f) (n-1)! có tốc độ tăng trưởng thấp hơn n! Vì (n-1)! = O(n!) và (n-1)! = **Ω** ((n-1)!)

(2) Xác định mối quan hệ độ phức tạp thuật toán theo kí pháp tiệm cận.

Câu 1.

a. **In best case**: O(1) vì số lần tính toán là hằng số 1 (50x+ 7 chỉ tính 1 lần)
b. **In average case:** O(1) vì số lần tính toán là hằng số 1 (50x+ 7 chỉ tính 1 lần)
c. **In worst case:** O(1) vì số lần tính toán là hằng số 1 (50x+ 7 chỉ tính 1 lần)

Câu 2.

a) **True** vì $2n(n -1)/2 = n^2 – n$ với $n > 1$ có tốc độ tăng trưởng $O(n^2) < O(n^3)$
b) **True** vì $2n(n -1)/2\ n^2 – n$ với $n > 1$ có tốc độ tăng trưởng $O(n^2) <= O(n^2)$
c) **False** vì $2n(n -1)/2$ có tốc độ tăng trưởng $O(n^2) < \Theta(n^3)$
d) **True** vì vì $2n(n -1)/2\ n^2 – n$ với $n > 1$ có tốc độ tăng trưởng n với n = 1

Câu 5.

Thứ tự tốc độ tăng trưởng từ thấp nhất đến cao nhất là:

$$\sqrt{n} < \ln(n)^3 < 2\lg(n + 50)^5 < 0.05n^{10} + 3n^3 + 1 < 3^{2n} < 3^{3n} < (n^2 + 3)!$$

(3) Thiết kế thuật toán, chứng minh tính đúng và xác định độ phức tạp của thuật toán.

Bài toán: Xây dựng thuật toán sắp xếp chọn theo ý tưởng sau: sắp xếp n số lưu trong mảng a bằng cách tìm phần tử nhỏ nhất của a và đổi nó với phần tử a[1]. Sau đó tìm phần tử nhỏ thứ hai của a, và đổi nó với a[2]. Tiếp tục với n − 1 phần tử của a.

- **Mã giả của thuật toán là:**

```
for i from 1 to n-1 do:
        minIndex = i
        for j from i + 1 to n-1 do:
                if a[j] < a[minIndex] then:
                        minIndex = j
        swap a[i]  with a[minIndex]
```

- **Vì sao chỉ cần thực hiện với n -1 phần tử đầu tiên**, thay cho cả n phần tử vì khi sắp xếp đến phần tử n -1 rồi thì vị trí cuối cùng (n) cũng đứng đúng vị trí của nó
- **Trường hợp tốt nhất là O(n^2)** (các phần tử đã sắp xếp theo thứ tự đúng) vì có 2 vòng lặp for, for from 1 to n – 1 và for from i+1 to n-1 và vẫn phải so sánh giữa các phần từ để tìm được phần tử nhỏ nhất trong mỗi lần lặp.
- **Trường hợp xấu nhất là O(n^2)** (các phần tử sắp xếp theo thứ tự ngược với thứ tự đúng) vì có 2 vòng lặp for, for from 1 to n – 1 và for from i+1 to n-1 và phải so sánh các phần tử với nhau trong mỗi lần lặp để tìm ra phần tử nhỏ nhất.
- **Bất biến vòng lặp** là 1 điều kiện logic được thiết lập được thiết lập trước khi thực hiện 1 vòng lặp và duy trì đúng sau mỗi lần lặp.
  - Bất biến vòng lặp:
    - Vòng lặp thứ i sẽ đưa phần tử nhỏ thứ i của mảng a về vị trí i.
    - Tại mỗi lần lặp i, mảng a[1...i-1] là đã được sắp xếp theo thứ tự tăng dần.
    - Mảng a[i...n] chứa các phần tử chưa được sắp xếp.
  - Khởi tạo:
    - Trước khi bắt đầu vòng lặp i, mảng a[1...i-1] chưa có phần tử nào.

    => Do đó tính chất khởi tạo được thỏa mãn.
  - Duy trì:
    - Tại mỗi lần lặp i, ta tìm phần tử nhỏ nhất trong mảng a[i...n] và đổi chỗ nó với a[i].
    - Sau đó, các phần tử trong mảng a[1...i] đã được sắp xếp theo thứ tự tăng dần.
    - Mảng a[i+1...n] chứa các phần tử chưa được sắp xếp.

    => Tính chất duy trì được thỏa mãn.

- Kết thúc:
  - Sau khi lặp n – 1 lần, mảng a[1...n-1] đã được sắp xếp theo thứ tự tăng dần.
  - Vòng lặp cuối cùng sẽ sắp xếp phần tử cuối cùng của mảng a, và mảng a[1...n] đã được sắp xếp theo thứ tự tăng dần.

  => Tính kết thúc được thỏa mãn.

Vậy thuật toán sắp xếp chọn là đúng đắn.